

# Full Stack Coding Challenge: Secure Task Management System

## Overview

Design and implement a secure **Task Management System** using **role-based access control (RBAC)** in a modular **NX monorepo**.

The system must allow users to manage tasks securely, ensuring only authorized users can access and modify data based on their roles and organizational hierarchy.

### Time Limit:

Most candidates spend around 8 hours on this assessment. We do not expect a fully complete solution.

Focus on correctness, security, and clear reasoning over feature completeness or polish.

**Document any tradeoffs or unfinished areas in your README.**

---

## Monorepo Structure (NX Workspace)

### Repository Naming

Please name your repository with your first name's first letter, last name, a hyphen (-) and a randomly generated uuid.

### Example:

John Doe → `jdoe-0a19fc14-d0eb-42ed-850d-63023568a3e3`

### Workspace Layout

```
apps/
  api/      → NestJS backend
  dashboard/ → Angular frontend

libs/
  data/     → Shared TypeScript interfaces and DTOs
  auth/     → Reusable RBAC logic and decorators
```

---

# Core Features

## Backend (NestJS + TypeORM + SQLite/PostgreSQL)

### Data Models

- Users
- Organizations (2-level hierarchy)
- Roles: Owner, Admin, Viewer
- Permissions
- Tasks (resource)

### Access Control Logic

- Implement decorators and guards for access checks
- Enforce ownership and organization-level access
- Implement role inheritance logic
- Scope task visibility based on role
- Implement basic audit logging (console or file)

### API Endpoints

- `POST /tasks`  
Create task (with permission check)
- `GET /tasks`  
List accessible tasks (scoped to role and organization)
- `PUT /tasks/:id`  
Edit task (if permitted)
- `DELETE /tasks/:id`  
Delete task (if permitted)
- `GET /audit-log`  
View access logs (Owner and Admin only)

### Authentication Requirements

- Do **not** use mock authentication
  - Implement real authentication using **JWT**
  - Authenticate via login and include token in all requests
  - Include token verification middleware or guards on all endpoints
-

## Frontend (Angular + TailwindCSS)

### Task Management Dashboard

- Create, edit, and delete tasks
- Sort, filter, and categorize tasks (e.g., Work, Personal)
- Drag-and-drop for task reordering or status changes
- Fully responsive design (mobile to desktop)

### Authentication UI

- Login UI that authenticates against the backend
- Store JWT after login
- Attach JWT to all API requests

### State Management

- Use any state management solution of your choice
- 

## Bonus Features (Optional)

- Task completion visualization (e.g., bar chart)
  - Dark/light mode toggle
  - Keyboard shortcuts for task actions
- 

## Testing Strategy

### Backend

- Use **Jest**
- Test RBAC logic, authentication, and API endpoints

### Frontend

- Use **Jest** or **Karma**
  - Test components and state management logic
-

# **README Requirements (Must Include)**

## **Setup Instructions**

- How to run backend and frontend applications
- `.env` configuration (JWT secrets, database config)

## **Architecture Overview**

- NX monorepo layout and rationale
- Explanation of shared libraries and modules

## **Data Model Explanation**

- Schema description
- ERD or diagram

## **Access Control Implementation**

- Role, permission, and organization hierarchy
- How JWT authentication integrates with access control

## **API Documentation**

- Endpoint list
- Sample requests and responses

## **Future Considerations**

- Advanced role delegation
  - Production-ready security:
    - JWT refresh tokens
    - CSRF protection
    - RBAC caching
  - Efficient scaling of permission checks
-

## **Video Guidelines**

The video should briefly walk through your solution, highlight key technical decisions (especially around authentication and RBAC), and explain any tradeoffs or unfinished areas.

- Length: Up to 10 minutes
- Format: .mp4 (preferred) or other common formats (.mov, .avi, .wmv, .mkv)
- File size (if uploading): Up to 300 MB
- Editing: No editing required. Focus on clarity and technical explanation.
- If sharing a link: Ensure “anyone with the link can view”

## **Evaluation Criteria**

- Secure and correct RBAC implementation
  - JWT-based authentication
  - Clean, modular NX architecture
  - Code clarity, structure, and maintainability
  - Responsive and intuitive UI
  - Test coverage
  - Documentation quality
  - Bonus for elegant UI/UX or advanced features
- 

## **Important Submission Instructions**

Submit your completed work through the official portal:

<https://forms.gle/1iJ2AHzMWsWeCLUE6>

This portal ensures proper tracking and routing to the hiring team